LEARN C++
Skill up with our free tutorials

0.9 — Configuring your compiler: Build configurations

ALEX[1] OCTOBER 23, 2023

A **build configuration** (also called a **build target**) is a collection of project settings that determines how your IDE will build your project. The build configuration typically includes things like what the executable will be named, what directories the IDE will look in for other code and library files, whether to keep or strip out debugging information, how much to have the compiler optimize your program, etc… Generally, you will want to leave these settings at their default values unless you have a specific reason to change something.

When you create a new project in your IDE, most IDEs will set up two different build configurations for you: a release configuration, and a debug configuration.

The **debug configuration** is designed to help you debug your program, and is generally the one you will use when writing your programs. This configuration turns off all optimizations, and includes debugging information, which makes your programs larger and slower, but much easier to debug. The debug configuration is usually selected as the active configuration by default. We'll talk more about debugging techniques in a later lesson.

The **release configuration** is designed to be used when releasing your program to the public. This version is typically optimized for size and performance, and doesn't contain the extra debugging information. Because the release configuration includes all optimizations, this mode is also useful for testing the performance of your code (which we'll show you how to do later in the tutorial series).

When the *Hello World* program (from lesson 0.7 -- Compiling your first program (https://www.learncpp.com/cpp-tutorial/compiling-your-first-program/)[3]) was built using Visual Studio, the executable produced in the debug configuration was 65KB, whereas the executable built in the release version was 12KB. The difference is largely due to the extra debugging information kept in the debug build.

Although you can create your own custom build configurations, you'll rarely have a reason to unless you want to compare two builds made using different compiler settings.

Best practice

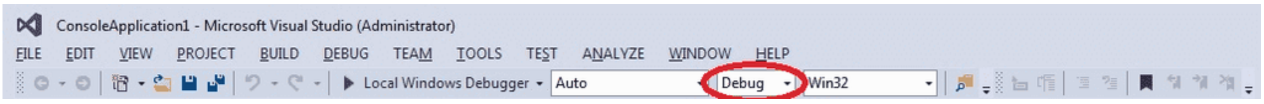
Use the *debug* build configuration when developing your programs. When you're ready to release your executable to others, or want to test performance, use the *release* build configuration.

Some IDEs (e.g. Visual Studio) also create separate build configurations for different platforms. For example, Visual Studio creates build configurations for both the x86 (32-bit) and the x64 (64-bit) platforms.

Switching between build configurations

For Visual Studio users

There are multiple ways to switch between *debug* and *release* in Visual Studio. The easiest way is to set your selection directly from the *Solution Configurations* dropdown in the *Standard Toolbar Options*:



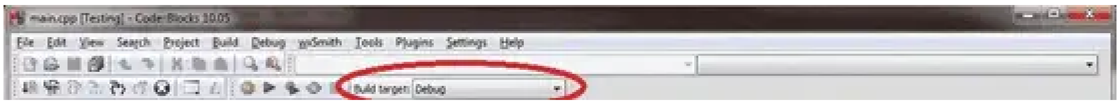
Set it to *Debug* for now.

You can also access the configuration manager dialog by selecting *Build menu > Configuration Manager*, and change the *active solution configuration*.

To the right of the *Solutions Configurations* dropdown, Visual Studio also has a *Solutions Platform* dropdown that allows you to switch between x86 (32-bit) and x64 (64-bit) platforms.

For Code::Blocks users

In Code::Blocks, you should see an item called *Build Target* in the *Compiler toolbar*:



Set it to *Debug* for now.

For GCC/G++ users

Add `-ggdb` to the command line when debugging and `-O2 -DNDEBUG` for release builds. Use the former for now.

For VS Code users

When you first ran your program, a new file called *tasks.json* was created under the *.vscode* folder in the explorer pane. Open the *tasks.json* file, find *“args”*, and then locate the line *“${file}”* within that section.

Above the *“${file}”* line, add a new line containing the following command (one per line) when debugging:

`"-ggdb",`

Above the *"${file}"* line, add new lines containing the following commands (one per line) for release builds:
`"-O2",`
`"-DNDEBUG",`

Modifying build configurations

In the next few lessons, we'll show you how to tweak some settings in your build configurations. Whenever changing a project setting, we recommend making the change in all build configurations.

This will help prevent making the change to the wrong build configuration, and ensure the change is still applied if you happen to switch build configurations later.

Tip

Whenever you update your project settings, make the change for all build configurations (unless it's not appropriate for some reason).

Next lesson
0.10 Configuring your compiler: Compiler extensions

4

Back to table of contents

5

Previous lesson
0.8 A few common C++ problems

6

7

Leave a comment...

Name*

Email*

Notify me about replies:

POST COMMENT

Find a mistake? Leave a comment above!

Avatars from https://gravatar.com/[10] are connected to your provided email address.

172 COMMENTS

Newest

Mona

April 5, 2024 1:46 am

I am using vscode on ubuntu. I can compile and build my program, however i cant find my task.json file,
~/.vscode $ ls
argv.json cli extensions

0 Reply

firildaqchi

February 8, 2024 9:04 am

>Add -ggdb to the command line when debugging and -O2 -DNDEBUG for release builds. Use the former for now.

Normally, I would run "g++ -o HelloWorld HelloWorld.cpp" and it worked. What is the use of -o flag?

My main question is, how do I run with -ggdb? when I do "g++ -ggdb HelloWorld HelloWorld.cpp" I get error:
/usr/bin/ld: cannot find HelloWorld: No such file or directory
collect2: error: ld returned 1 exit status

What is the reason?
Thanks in advance

2 Reply

Harry
Reply to firildaqchi [11] February 14, 2024 6:04 pm

g++ -ggdb HelloWorld.cpp -o HelloWorld

You need the option -o and flip the source first and then the executable file name with option -o

0 Reply

IceFloe
February 7, 2024 5:39 pm

Thank you for the lesson, thank you for preparing us in advance to work with the debugger, I have heard that many programmers ignore this, releasing their programs immediately in the release version and testing it without checking it for errors. So far I understand little how the debugger works, but I hope they will devote time to this in future lessons

0 Reply

S.E.
January 24, 2024 12:06 am

Two questions:

1.) Being that optimizations are removed for a debug configuration, are there rare scenarios where a bug only seems to appear in a debug version however doesn't seem to be replicable in the release version (or vice versa)?

2.) In the comments, I am seeing a lot of mention of a task.json, though I am not seeing any reference to it in the actual lesson here. Is it possible these comments are in relation to an outdated version of this page?

I appreciate your work. Thank you!

0 Reply

Alex Author
Reply to S.E. [12] January 25, 2024 3:14 pm

1. It's common for issues to appear in release but not debug. This is almost always attributable to doing something that produces undefined behavior, and the behavior is manifesting differently in the different builds.
2. All of the task.json stuff is in relationship to VS Code. I told people it's hard to configure but people apparently don't believe me. :)

2 Reply

WanderingGoose
January 23, 2024 12:38 pm

Do we need to build debug configurations on Eclipse? If so, how? I don't want to mess something up that I won't be able to fix later.

0 Reply

Atapa
January 14, 2024 3:05 pm

my task.json on vs code file looks like:

```
{
    "version": "2.0.0",
    "tasks": [
        {
            "type": "cppbuild",
            "label": "C/C++: clang++ build active file",
            "command": "/usr/bin/clang++",
            "args": [
                "-fcolor-diagnostics",
                "-fansi-escape-codes",
                "-ggdb",
                "-pedantic-errors",
                "-Wall",
                "-Weffc++",
                "-Wextra",
                "-Wconversion",
                "-Wsign-conversion",
                "-Werror",
                "-std=c++17",
                "${file}",
                "-o",
                "${fileDirname}/${fileBasenameNoExtension}"
            ],
            "options": {
                "cwd": "${fileDirname}"
            },
            "problemMatcher": [
                "$gcc"
            ],
            "group": {
                "kind": "build",
                "isDefault": true
            },
            "detail": "compiler: /usr/bin/clang++"
        }
    ]
}
```

This is my task.json file after following all instructions in this chapter. Is everything configured correctly?

EDIT: I prematurely commented on this section :)

Last edited 2 months ago by Atapa

0 Reply

Ali

Reply to Atapa 13 March 3, 2024 8:38 am

no idea

```
{
    "tasks": [
        {
            "type": "cppbuild",
            "label": "C/C++: g++ build active file",
            "command": "/usr/bin/g++",
            "args": [
                "-fdiagnostics-color=always",
                "-g",
                "-ggdb",
                "-pedantic-errors",
                "-Wall",
                "-Weffc++",
                "-Wextra",
                "-Wconversion",
                "-Wsign-conversion",
                "-Wshadow=global",
                "-Wshadow=local",
                "-Wshadow=compatible-local",

                "-Werror", // force to fix warning

                "-std=c++2a",
                // "${file}",

                "${fileDirname}/**.cpp",
                "-o",
                "${fileDirname}/${fileBasenameNoExtension}"
            ],
            "options": {
                "cwd": "${fileDirname}"
            },
            "problemMatcher": [
                "$gcc"
            ],
            "group": {
                "kind": "build",
                "isDefault": true
            },
            "detail": "Task generated by Debugger."
        }
    ],
    "version": "2.0.0"
}
```

Last edited 1 month ago by Ali

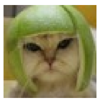
0 Reply

syon-k
January 10, 2024 7:07 am

can someone explain to me what "all build configurations" means and thanks.

Last edited 2 months ago by syon-k

0 Reply



Alex Author
Reply to syon-k [14] January 11, 2024 1:23 pm

It means all of the currently defined build configurations (e.g. release and debug and any others you've created).

0 Reply

Daz M
December 20, 2023 3:57 pm

I get the feeling that using CL.exe from VS Code in Windows is not covered in this.
both:
-ggdb
-pedantic-errors
get the ignored response when building:
cl : Command line warning D9002 : ignoring unknown option '-ggdb'
cl : Command line warning D9002 : ignoring unknown option '-pedantic-errors'

0 Reply

Harry
Reply to Daz M [15] February 14, 2024 6:10 pm

For MSVC cl.exe debug option is /Z7

Example:

cl /EHsc /Z7 basiccx.cpp /link /SUBSYSTEM:CONSOLE

Here is the list of cl.exe option on Microsoft website

https://learn.microsoft.com/en-us/cpp/build/reference/compiler-options-listed-alphabetically?view=msvc-170

Last edited 1 month ago by Harry

0 Reply

Daz M

Reply to Harry[16] February 14, 2024 8:02 pm

Thank you :)

0 Reply

AbeerOrTwo

December 13, 2023 6:56 pm

build config

0 Reply

anon-kun

November 24, 2023 12:13 pm

If you're using VS Code on Windows and you get a segmentation fault with the simple hello world problem, check if your default terminal profile is Git Bash and change it to Windows' command prompt. WSL also works fine.

0 Reply

Links

1. https://www.learncpp.com/author/Alex/
2. https://ikeamenu.com/humix/video/Ipms8sWsckf
3. https://www.learncpp.com/cpp-tutorial/compiling-your-first-program/
4. https://www.learncpp.com/cpp-tutorial/configuring-your-compiler-compiler-extensions/
5. https://www.learncpp.com/
6. https://www.learncpp.com/cpp-tutorial/a-few-common-cpp-problems/
7. https://www.learncpp.com/configuring-your-compiler-build-configurations/
8. https://www.learncpp.com/cpp-tutorial/void/
9. https://www.learncpp.com/cpp-tutorial/constant-variables-named-constants/
10. https://gravatar.com/
11. https://www.learncpp.com/cpp-tutorial/configuring-your-compiler-build-configurations/#comment-593418

12. https://www.learncpp.com/cpp-tutorial/configuring-your-compiler-build-configurations/#comment-592787
13. https://www.learncpp.com/cpp-tutorial/configuring-your-compiler-build-configurations/#comment-592377
14. https://www.learncpp.com/cpp-tutorial/configuring-your-compiler-build-configurations/#comment-592134
15. https://www.learncpp.com/cpp-tutorial/configuring-your-compiler-build-configurations/#comment-591233
16. https://www.learncpp.com/cpp-tutorial/configuring-your-compiler-build-configurations/#comment-593640